北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

TensorFlow平台简介

郑允波 李雪 任岩 刘文聪 吴迪

C目录
CONTENTS

北京理工大學

平台简介

平台简介

TensorFlow

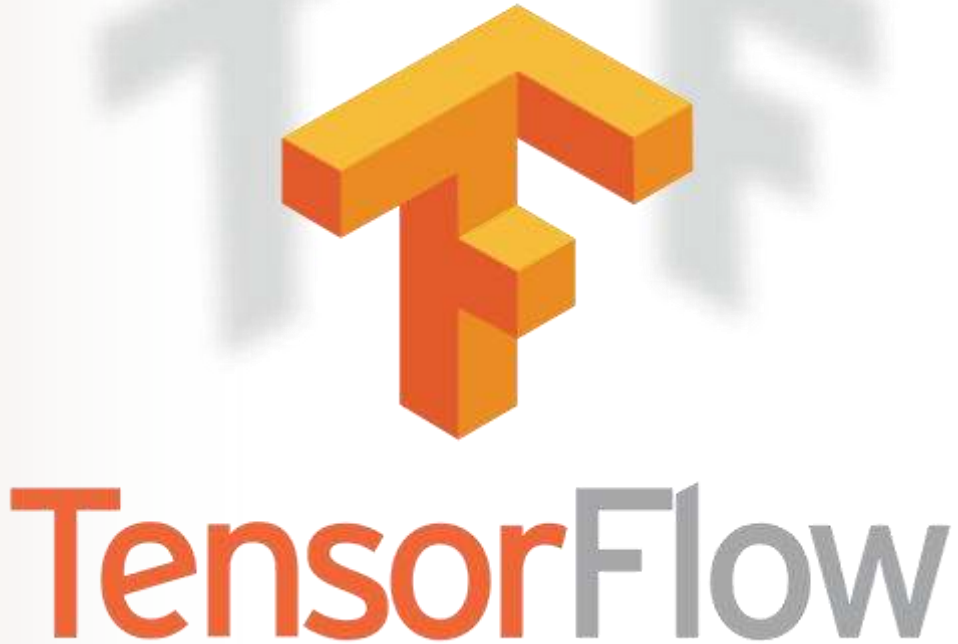
2.0 Is *Here*!

平台简介

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

TensorFlow

TensorFlow用于各种感知和语言理解任务的机器学习的开源软件库。由谷歌大脑团队开发，于2015年11月9日在Apache 2.0开源许可证下发布。是谷歌第一代分布式机器学习框架DistBelief的继承产品。



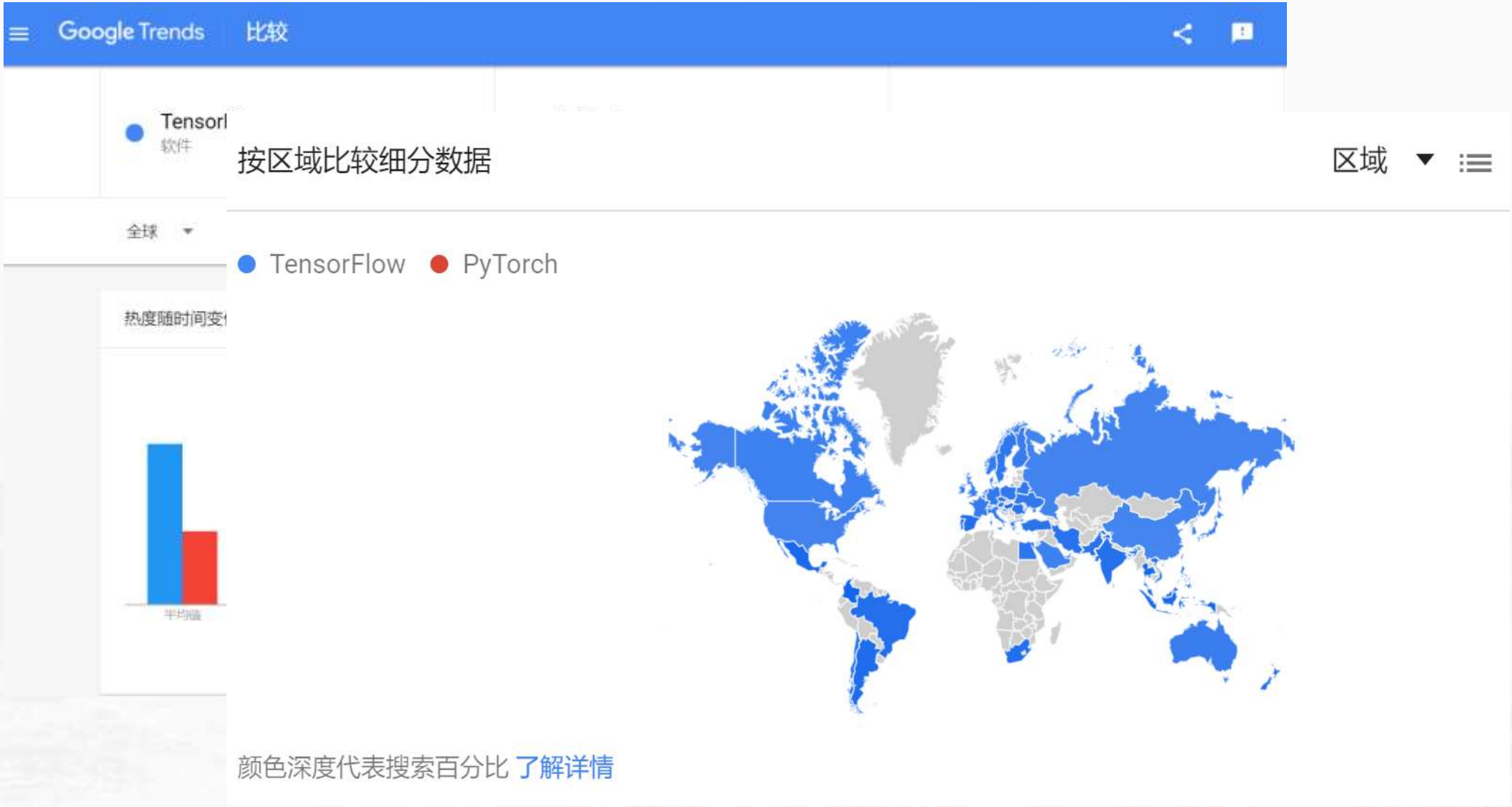
平台简介

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

2015年	2016年	2017年	2018年	2019年
11月9日开源，成为2015年最受关注的开源项目之一。	将DeepMind模型迁移到TensorFlow，并改进了对移动设备的支持。	v1.0版本发布，增加了专用的编译器和调试工具，同时引入高级API。	与Nvidia合作实现了对GPU硬件计算环境的高度优化，更新或添加了新的API。	v2.0版本发布，与Keras紧密集成，支持高性能分布训练，优化GPU训练性能

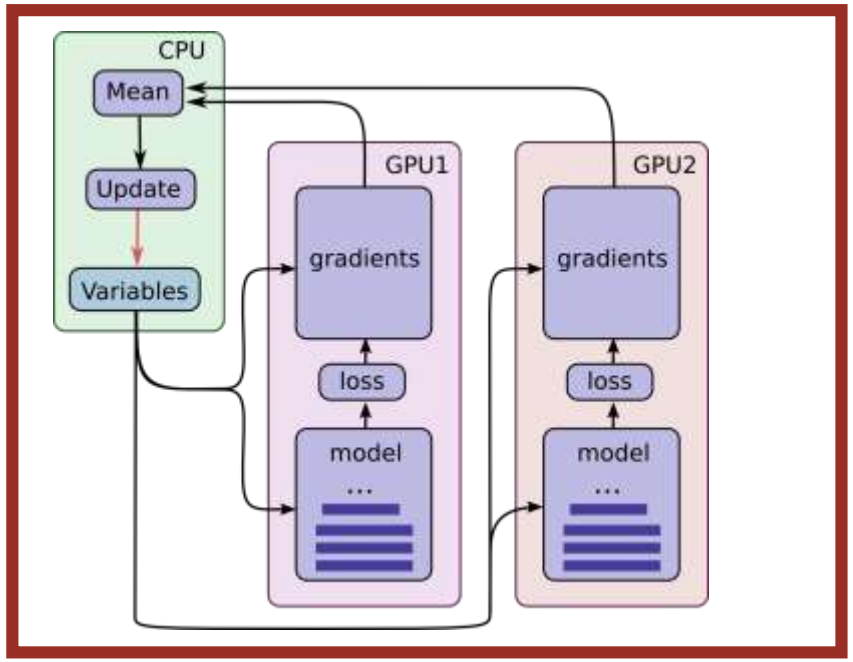
平台简介

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY



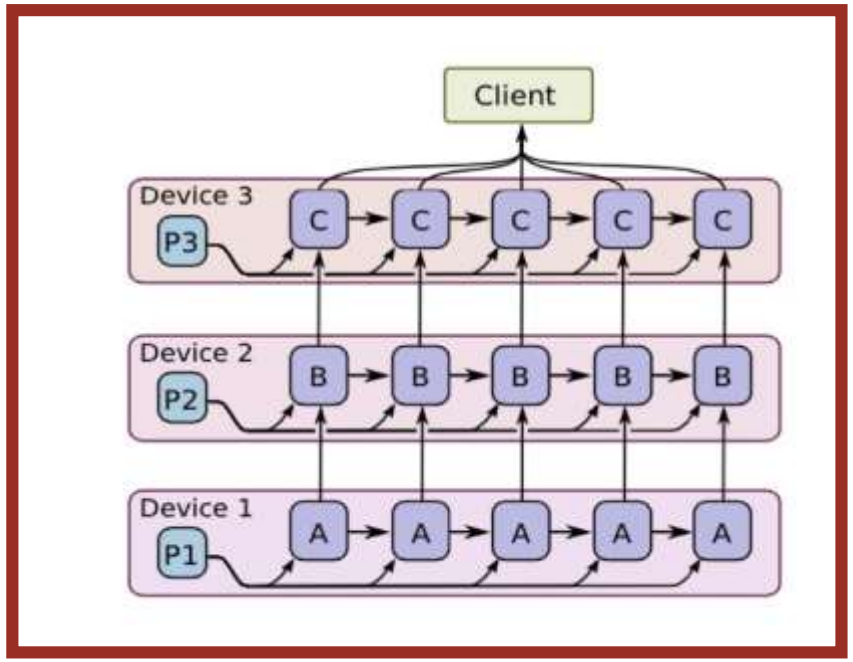
平台特点

平台特点



性能最优化

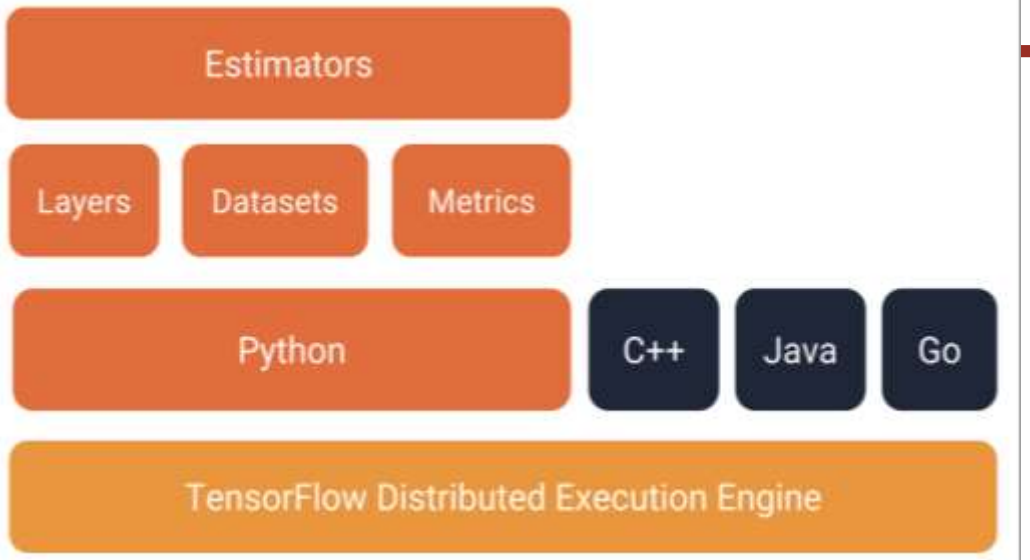
TensorFlow 给予了线程、队列、异步操作等以最佳的支持，可以将你手边硬件的计算潜能全部发挥出来。



性能最优化

对模型进行切分，让模型的不同部分执行在不同的设备上，这样可以一个迭代的样本可以在不同的设备上同时执行。

平台特点

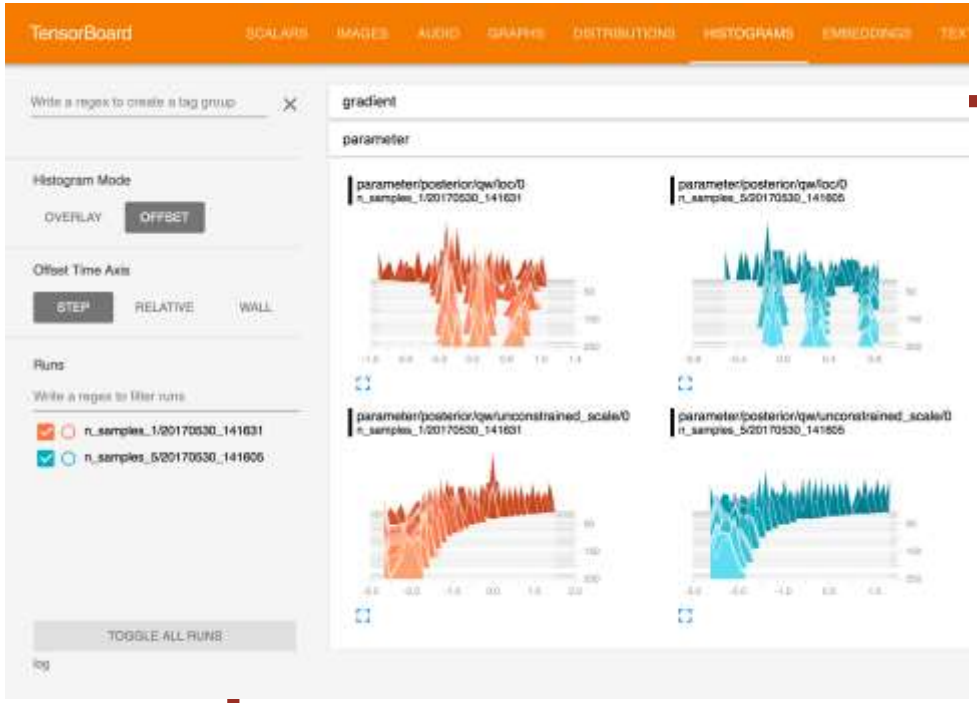


多语言支持

TensorFlow设计了高中低不同等级的API，多语言的兼容性使得开发者可以尽可能的使用自己熟悉的开发语言。

平台特点

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

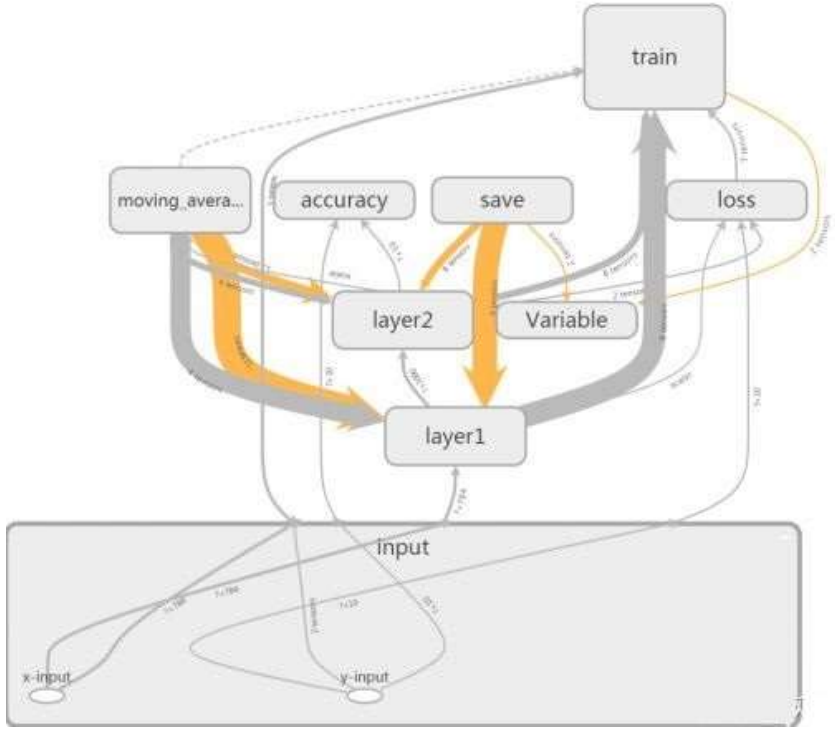


可视化

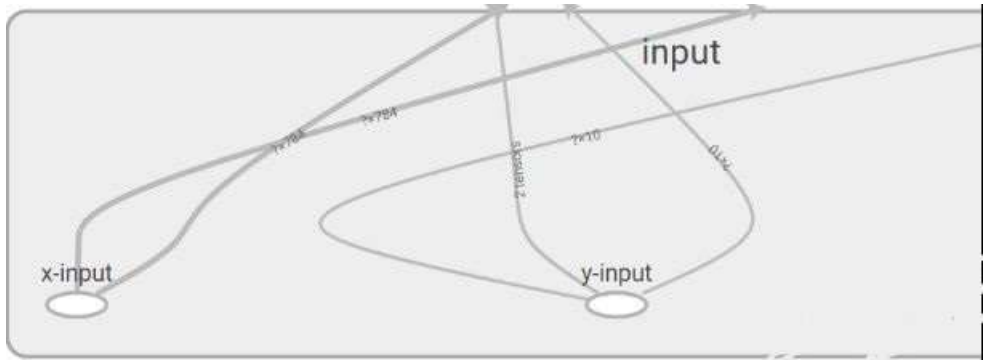
TensorFlow 的 Tensorboard 是个非常棒的功能。它内置在 TensorFlow 中，在调试和比较不同的训练状况时非常有用。

平台特点

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY



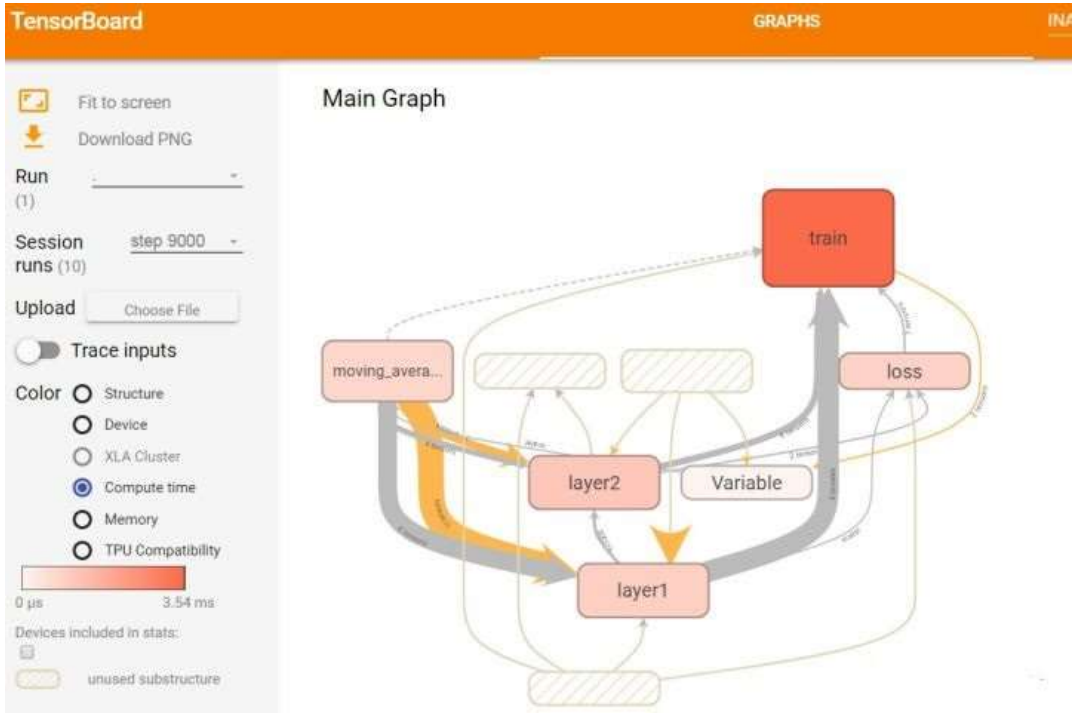
显示结点信息



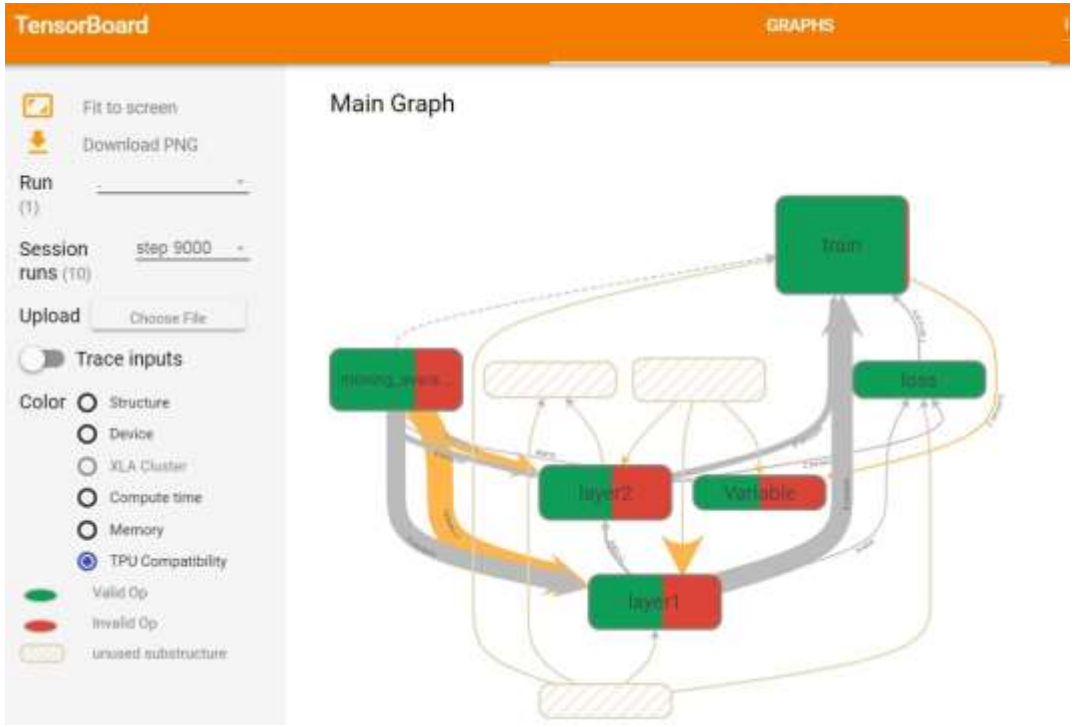
放大可看更详细的信息

平台特点

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY



不同迭代数量后的变化

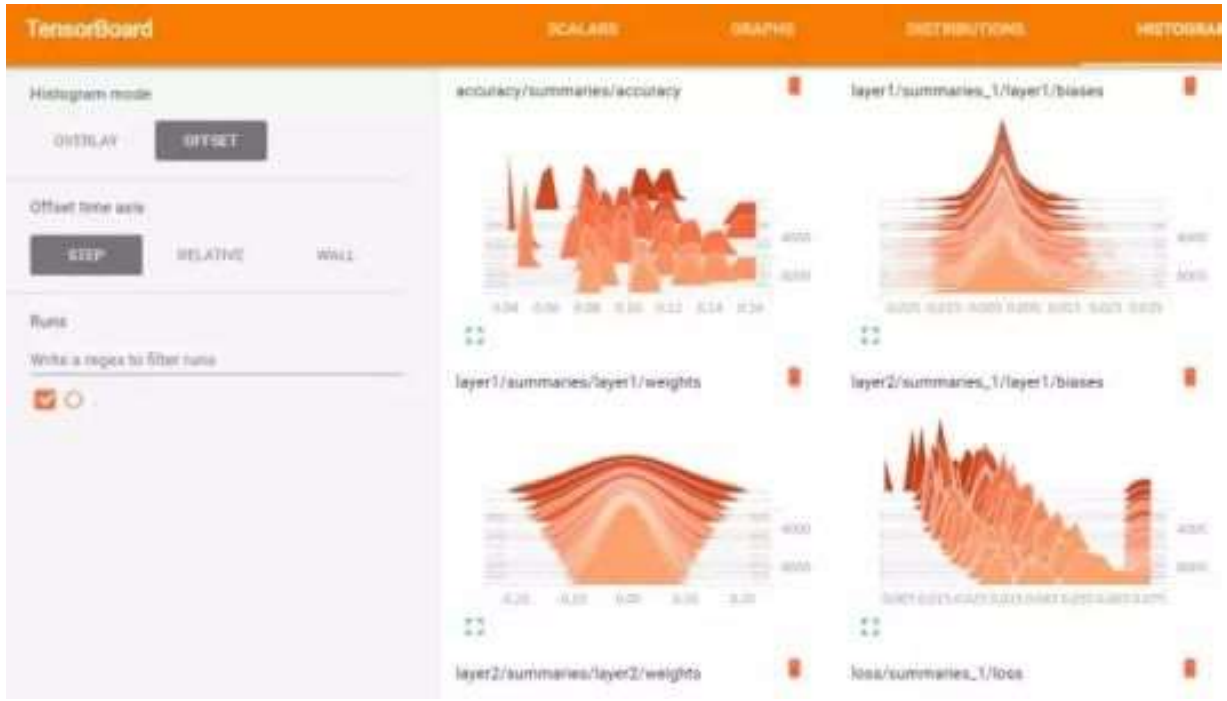


结点运行的设备信息

平台特点

北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

随迭代标量数据的变化



随迭代张量信息的变化

平台特点

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

真正的可移植性

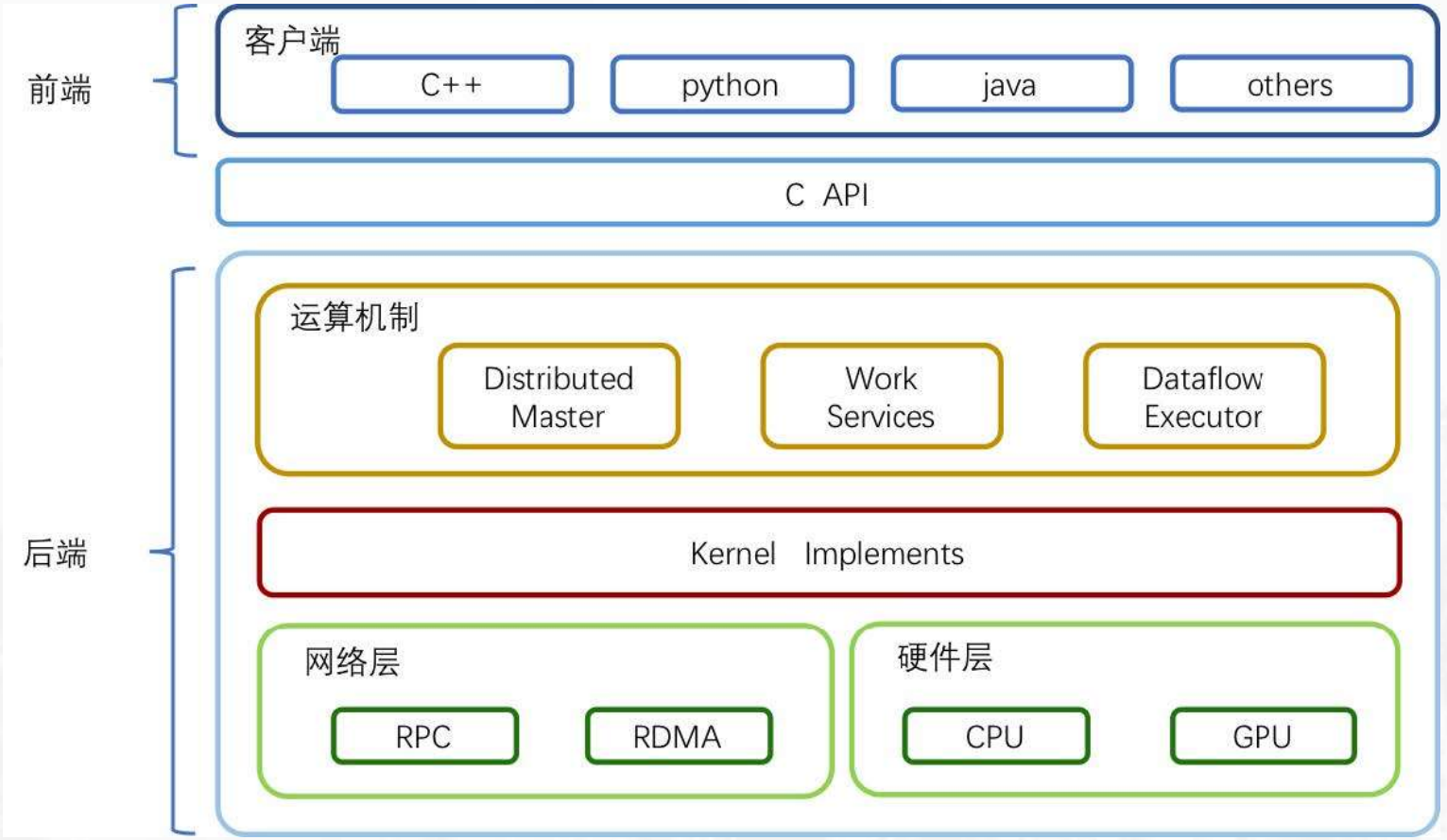
TensorFlow可以运行在台式机、服务器、手机移动设备等上，可以把你的训练好的模型作为产品的一部分应用到移动端设备上。



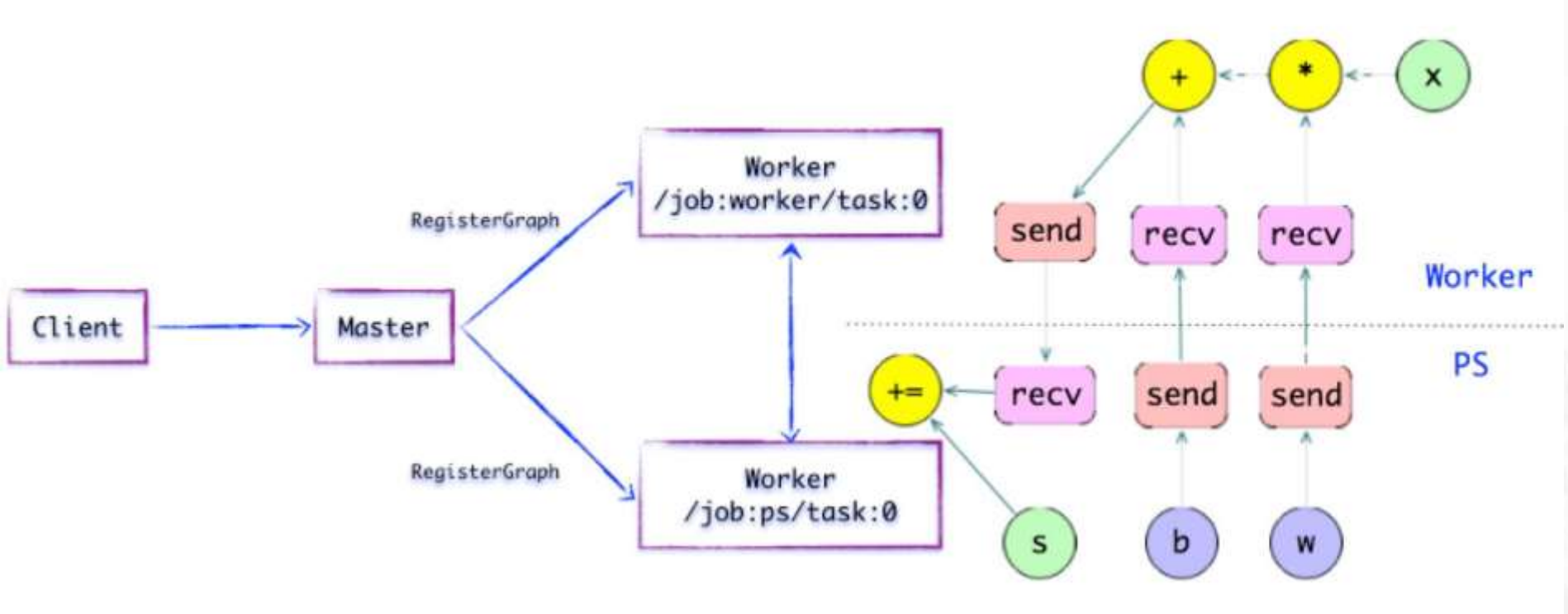

技术原理

TensorFlow的系统架构

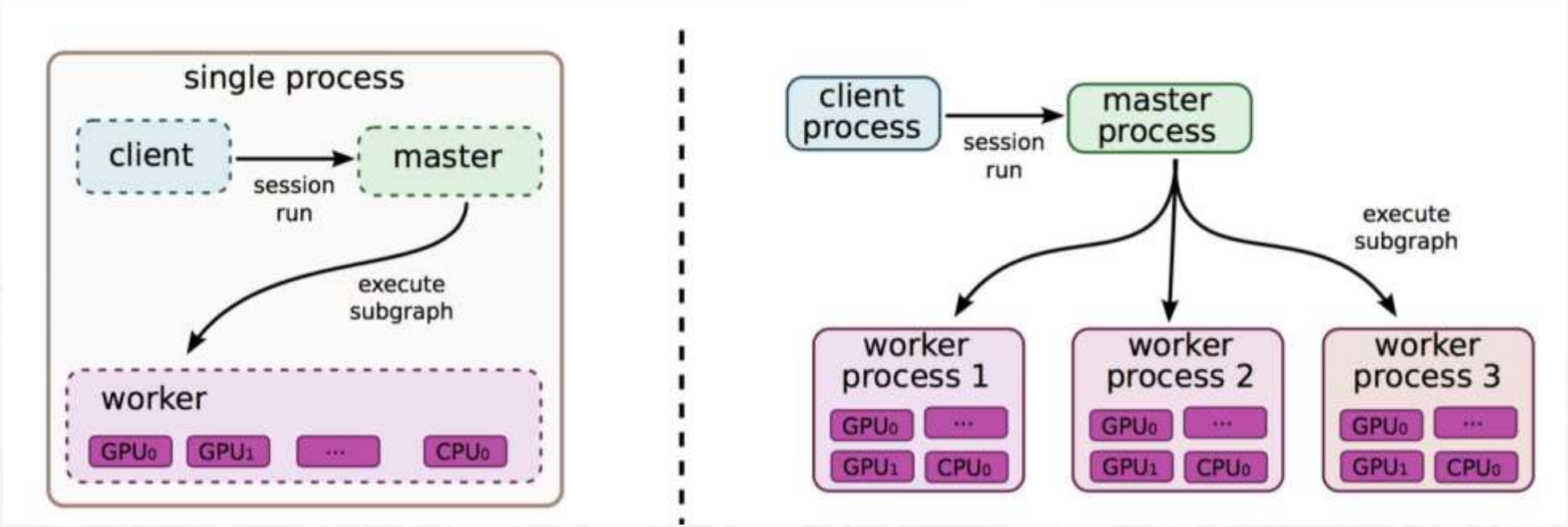
层	功能	组件
视图层	计算图可视化	TensorFlow
工作流层	数据集准备、存储、加载	Keras/TF Slim
计算图层	计算图构造与优化，前/后向计算	TensorFlow Core
高维计算层	高维数组处理	Eigen
数值计算层	矩阵计算/卷积计算	BLAS/cuBLAS/cuRAND/cuDNN
网络层	通信	Grpc/RDMA
设备层	硬件	CPU/GPU



TensorFlow的技术架构



TF组件的交互，master节点给两种类型的节点分发任务：
/job:ps/task:0: 负责模型参数的存储和更新
/job:worker/task:0:负责模型的训练和推理



master，用来与客户端交互，同时调度任务

worker process，工作节点，每个worker process可以访问一到多个device。

实例展示

验证码识别

喵汪识别

验证码识别

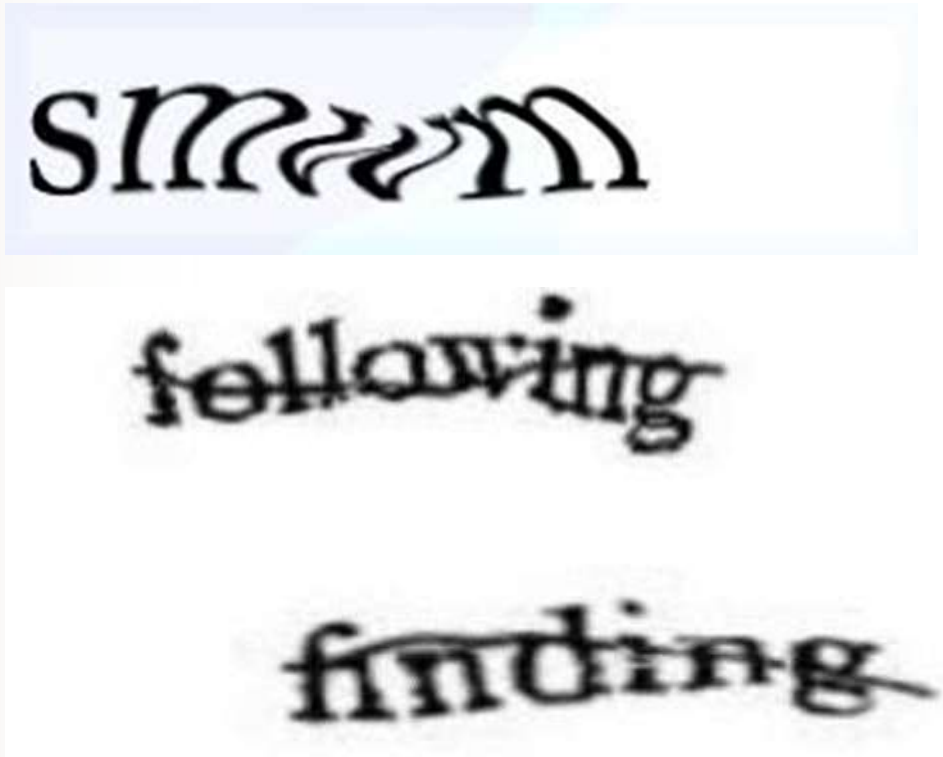
验证码识别

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

验证码

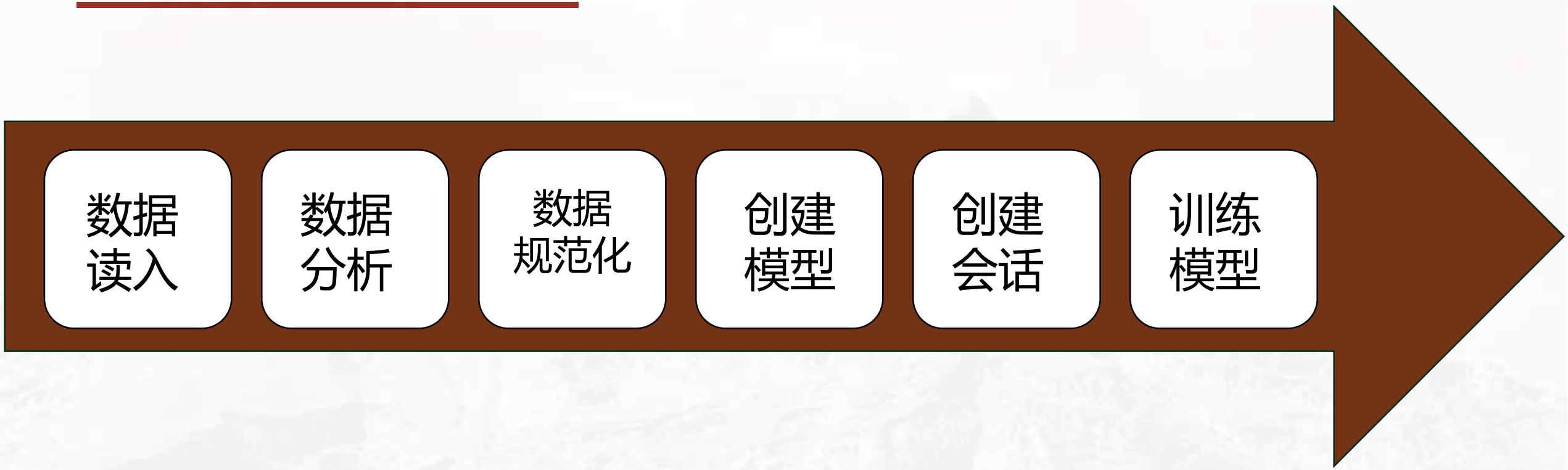
全自动区分计算机和人类的公开图灵测试（CAPTCHA），俗称验证码，是一种区分用户是计算机或人的公共全自动程序，一种常用的CAPTCHA测试是让用户输入一个扭曲变形的图片上所显示的文字或数字。

利用TensorFlow平台，通过验证码数据集对模型进行训练，模型输出为验证码识别结果。



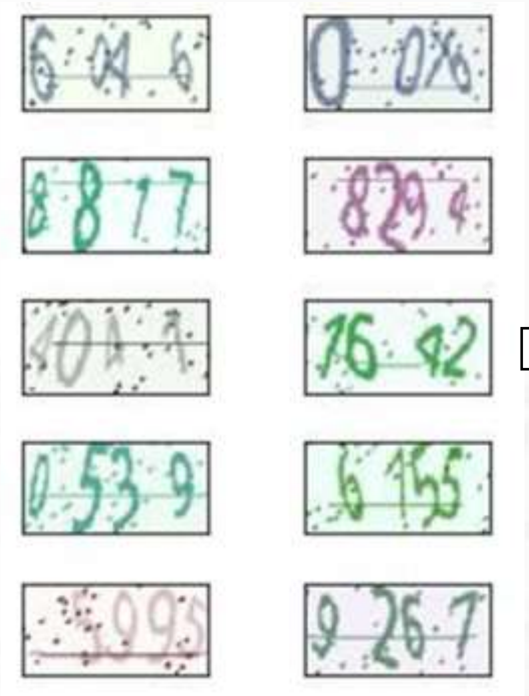
算法流程

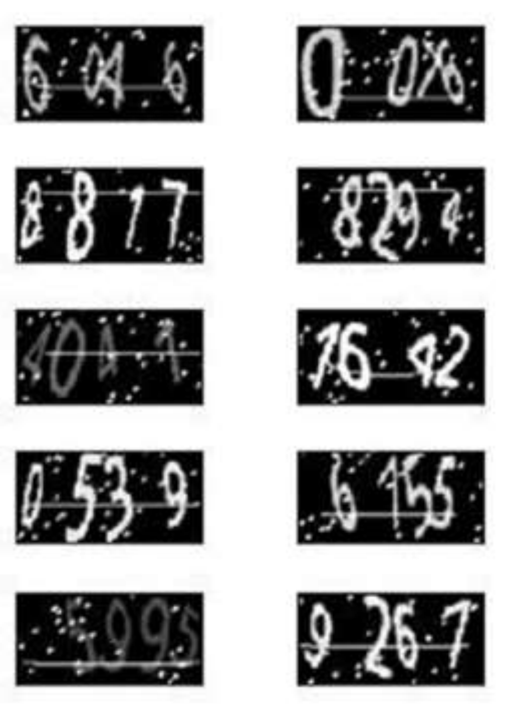
北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY



利用Catpcha库，生成9999张验证码和标签用作训练集，并生成100张验证码及标签用作测试集.

输入数据规范化





```
array([[0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.32744314, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
        0.98508627, 0.98508627, 0.98508627, 0.98508627, 0.98508627,
```

RGB图 -> 灰度图 -> 规范化数据

将彩色图像转化为只含灰度数据的多维数组

输出数据规范化


```
array([0., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
       1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 1., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 1., 0., 0., 0.])
```

One-hot 编码：

输出数据为向量形式，选取每行向量中概率最高的数字为识别出来的验证码数字

模型结构

三层卷积网络：
利用卷积网络进行特征提取

四个全连接网络：
分别对四个验证码数字进行预测，
求得损失函数进行后向传播

输出数据规范化

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

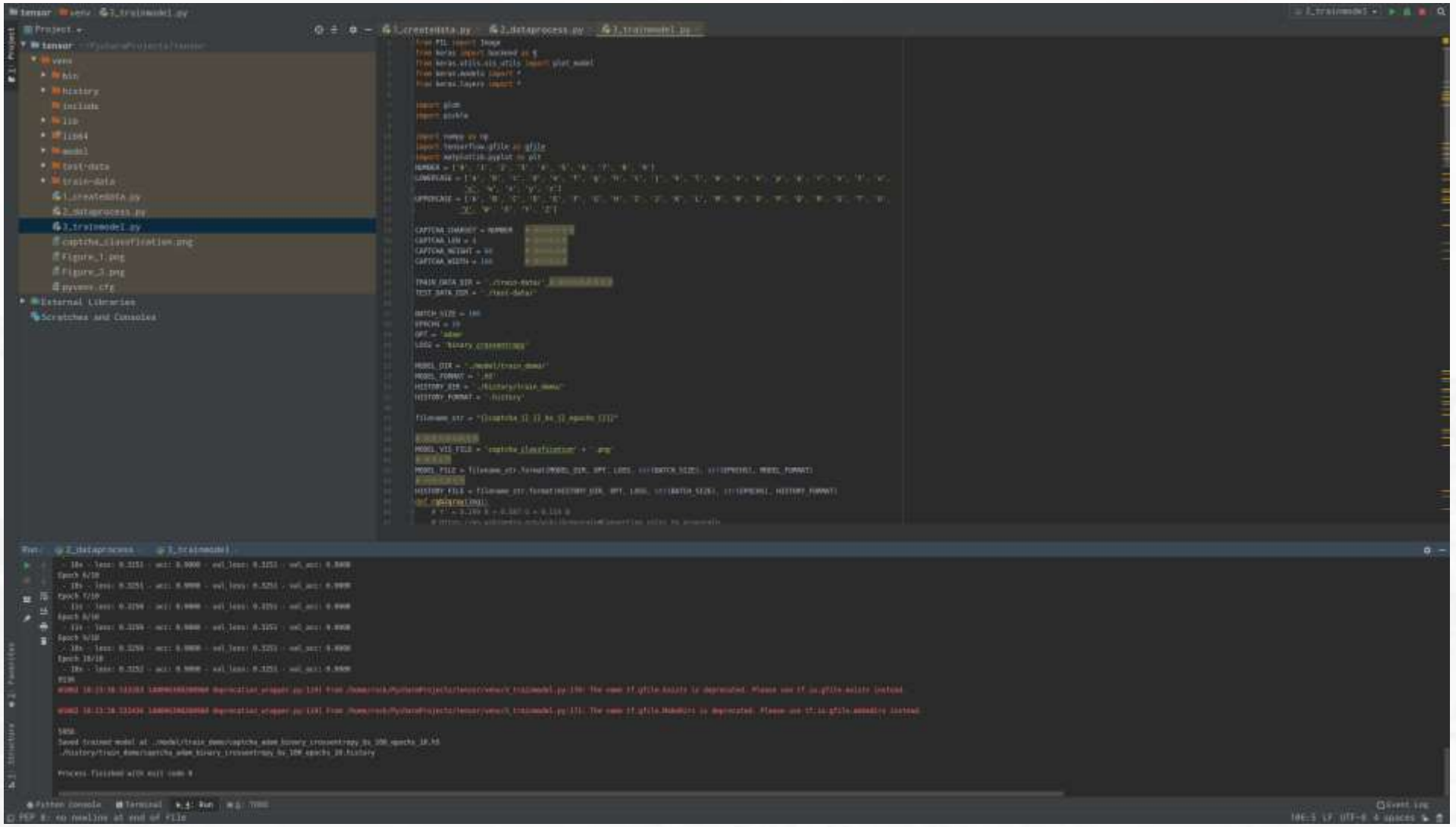
```python
# 输入层
inputs = Input(shape = input_shape, name = "inputs")

# 第1层卷积
conv1 = Conv2D(32, (3, 3), name = "conv1")(inputs)
relu1 = Activation('relu', name="relu1")(conv1)

# 第2层卷积
conv2 = Conv2D(32, (3, 3), name = "conv2")(relu1)
relu2 = Activation('relu', name="relu2")(conv2)
pool2 = MaxPooling2D(pool_size=(2,2), padding='same', name="pool2")(relu2)

# 第3层卷积
conv3 = Conv2D(64, (3, 3), name = "conv3")(pool2)
relu3 = Activation('relu', name="relu3")(conv3)
pool3 = MaxPooling2D(pool_size=(2,2), padding='same', name="pool3")(relu3)

# 将 Pooled feature map 摊平后输入全连接网络
x = Flatten()(pool3)

# Dropout
x = Dropout(0.25)(x)

# 4个全连接层分别做10分类，分别对应4个字符。
x = [Dense(10, activation='softmax', name='fc%d'%(i+1))(x) for i in range(4)]

# 4个字符向量拼接在一起，与标签向量形式一致，作为模型输出。
outs = Concatenate()(x)

# 定义模型的输入与输出
model = Model(inputs=inputs, outputs=outs)
model.compile(optimizer=OPT, loss=LOSS, metrics=['accuracy'])
```

三层卷积网络

四个全连接网络

实际训练过程

北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY



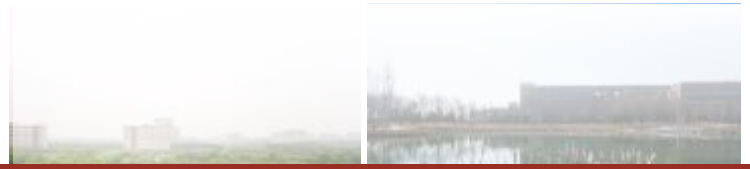
训练结果

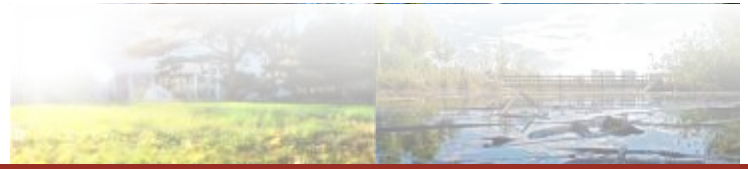
北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

```
: history = model.fit(X_train,
                      Y_train,
                      batch_size=BATCH_SIZE,
                      epochs=EPOCHS,
                      verbose=2,
                      validation_data=(X_test, Y_test))

Train on 3956 samples, validate on 954 samples
Epoch 1/10
 - 149s - loss: 0.3270 - acc: 0.9000 - val_loss: 0.3247 - val_acc: 0.9000
Epoch 2/10
 - 122s - loss: 0.3229 - acc: 0.9000 - val_loss: 0.3195 - val_acc: 0.9000
Epoch 3/10
 - 114s - loss: 0.2987 - acc: 0.9004 - val_loss: 0.2726 - val_acc: 0.9028
Epoch 4/10
 - 106s - loss: 0.2257 - acc: 0.9164 - val_loss: 0.2303 - val_acc: 0.9171
Epoch 5/10
 - 103s - loss: 0.1799 - acc: 0.9337 - val_loss: 0.2171 - val_acc: 0.9209
Epoch 6/10
 - 113s - loss: 0.1523 - acc: 0.9447 - val_loss: 0.2062 - val_acc: 0.9254
Epoch 7/10
 - 112s - loss: 0.1383 - acc: 0.9498 - val_loss: 0.2048 - val_acc: 0.9260
Epoch 8/10
 - 127s - loss: 0.1251 - acc: 0.9550 - val_loss: 0.2052 - val_acc: 0.9260
Epoch 9/10
 - 161s - loss: 0.1144 - acc: 0.9587 - val_loss: 0.2013 - val_acc: 0.9285
Epoch 10/10
 - 159s - loss: 0.1063 - acc: 0.9618 - val_loss: 0.2045 - val_acc: 0.9268
```

测试结果：
损失函数值为0.1063
测试集的精度达到0.9618

实测结果：
10次验证只有2次正确，
模型仍需大量数据训练。





北京理工大學

喵汪识别

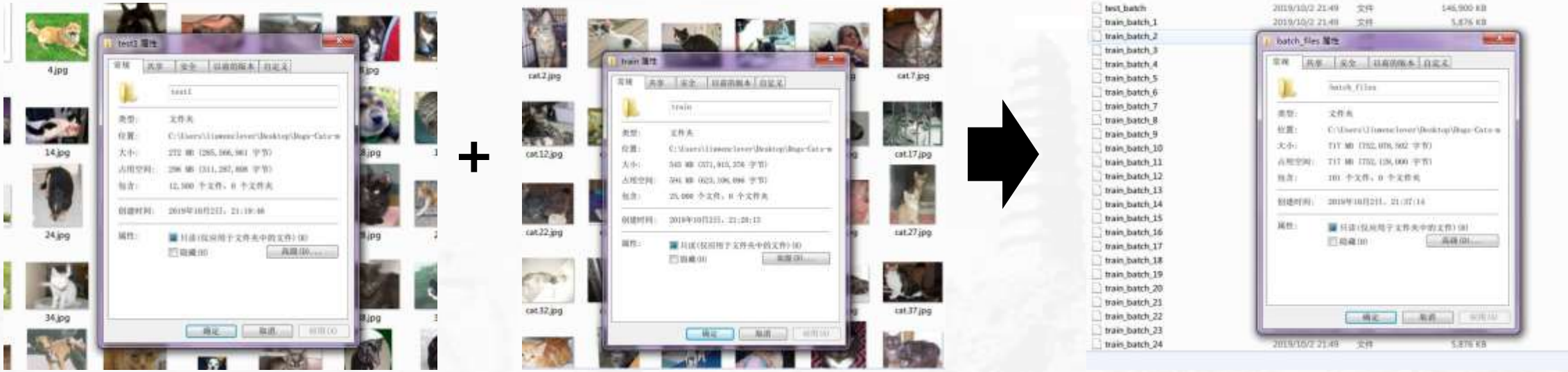
任务简介

对猫和狗的图片进行分类是深度学习图像分类的经典案例之一，猫和狗在外观上的差别还是挺明显的，无论是体型、四肢、脸庞或者毛发等等， 都是能通过肉眼很容易区分的。

通过基于TensorFlow平台实现的卷积神经网络，能够让机器来识别猫和狗。

算法流程

数据集的准备



猫狗照片的数据集直接从kaggle官网(https://www.kaggle.com/c/dogs-vs-cats)下载即可，下载后解压，可以看到有训练集和测试集，接着将图片转换为tensorflow中可以使用的批数据，即batch文件。

编写神经网络


```
conv1_1 = tf.layers.conv2d(x, 16, (3, 3), padding='same', activation=tf.nn.relu, name='conv1_1')
conv1_2 = tf.layers.conv2d(conv1_1, 16, (3, 3), padding='same', activation=tf.nn.relu, name='conv1_2')
pool1 = tf.layers.max_pooling2d(conv1_2, (2, 2), (2, 2), name='pool1')
conv2_1 = tf.layers.conv2d(pool1, 32, (3, 3), padding='same', activation=tf.nn.relu, name='conv2_1')
conv2_2 = tf.layers.conv2d(conv2_1, 32, (3, 3), padding='same', activation=tf.nn.relu, name='conv2_2')
pool2 = tf.layers.max_pooling2d(conv2_2, (2, 2), (2, 2), name='pool2')
conv3_1 = tf.layers.conv2d(pool2, 64, (3, 3), padding='same', activation=tf.nn.relu, name='conv3_1')
conv3_2 = tf.layers.conv2d(conv3_1, 64, (3, 3), padding='same', activation=tf.nn.relu, name='conv3_2')
pool3 = tf.layers.max_pooling2d(conv3_2, (2, 2), (2, 2), name='pool3')
conv4_1 = tf.layers.conv2d(pool3, 128, (3, 3), padding='same', activation=tf.nn.relu, name='conv4_1')
conv4_2 = tf.layers.conv2d(conv4_1, 128, (3, 3), padding='same', activation=tf.nn.relu, name='conv4_2')
pool4 = tf.layers.max_pooling2d(conv4_2, (2, 2), (2, 2), name='pool4')
flatten = tf.layers.flatten(pool4)
fc1 = tf.layers.dense(flatten, 512, tf.nn.relu)
fc1_dropout = tf.nn.dropout(fc1, keep_prob=keep_prob)
fc2 = tf.layers.dense(fc1, 256, tf.nn.relu)
fc2_dropout = tf.nn.dropout(fc2, keep_prob=keep_prob)
fc3 = tf.layers.dense(fc2, 2, None)
```

4个池化层 + 8个卷积层

3个全连接层

```
#定义占位符
self.x = tf.placeholder(tf.float32, [None, IMAGE_SIZE, IMAGE_SIZE, 3], 'input_data')
self.y = tf.placeholder(tf.int64, [None], 'output_data')
self.keep_prob = tf.placeholder(tf.float32)
#将图片输入到网络中
fc = self.conv_net(self.x, self.keep_prob)
#定义损失函数
self.loss = tf.losses.sparse_softmax_cross_entropy(labels=self.y, logits=fc)
self.y_ = tf.nn.softmax(fc)
#定义预测值
self.predict = tf.argmax(fc, 1)
#定义准确率
self.acc = tf.reduce_mean(tf.cast(tf.equal(self.predict, self.y), tf.float32))
self.train_op = tf.train.AdamOptimizer(LEARNING_RATE).minimize(self.loss)
self.saver = tf.train.Saver(max_to_keep=1)
```

模型的训练和测试（1万步）

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

```
acc_list = []
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for i in range(TRAIN_STEP):
        train_data, train_label, _ =
self.batch_train_data.next_batch(TRAIN_SIZE)

        eval_ops = [self.loss, self.acc, self.train_op]
        eval_ops_results = sess.run(eval_ops, feed_dict={
            self.x:train_data,
            self.y:train_label,
            self.keep_prob:0.7
        })
        loss_val, train_acc = eval_ops_results[0:2]

        acc_list.append(train_acc)
        if (i+1) % 100 == 0:
            acc_mean = np.mean(acc_list)

print('step:{0},loss:{1:.5},acc:{2:.5},acc_mean:{3:.5}'.format(
                i+1,loss_val,train_acc,acc_mean ))
```

```
 if (i+1) % 1000 == 0:
            test_acc_list = []
            for j in range(TEST_STEP):
                test_data, test_label, _ =
self.batch_test_data.next_batch(TRAIN_SIZE)
                acc_val = sess.run([self.acc],feed_dict={
                    self.x:test_data,
                    self.y:test_label,
                    self.keep_prob:1.0
                })
                test_acc_list.append(acc_val)
                print('[Test ] step:{0}, mean_acc:{1:.5}'.format(
                    i+1, np.mean(test_acc_list)
                ))
```

```
    # 保存训练后的模型
    os.makedirs(SAVE_PATH, exist_ok=True)
    self.saver.save(sess, SAVE_PATH +
'my_model.ckpt')
```

识别和分类

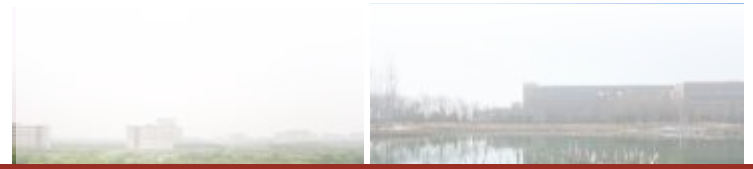
北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

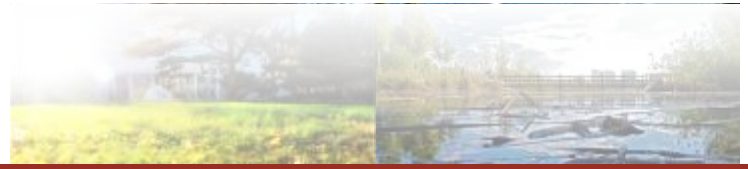
使用自己训练好的模型，把官网的测试图片(共12500张)识别后进行分类，并将分类后的图片分别写入到两个文件夹中

测试数据集

训练结果

可以看出，分类之后还有少数分类不对的结果。看来模型还有待提升，可以通过调整网络参数、调整学习率与学习步数、使用图像增强等技术对模型识别准确率进行提高。





北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

德以明理
学以精工

THANKS

第一小组